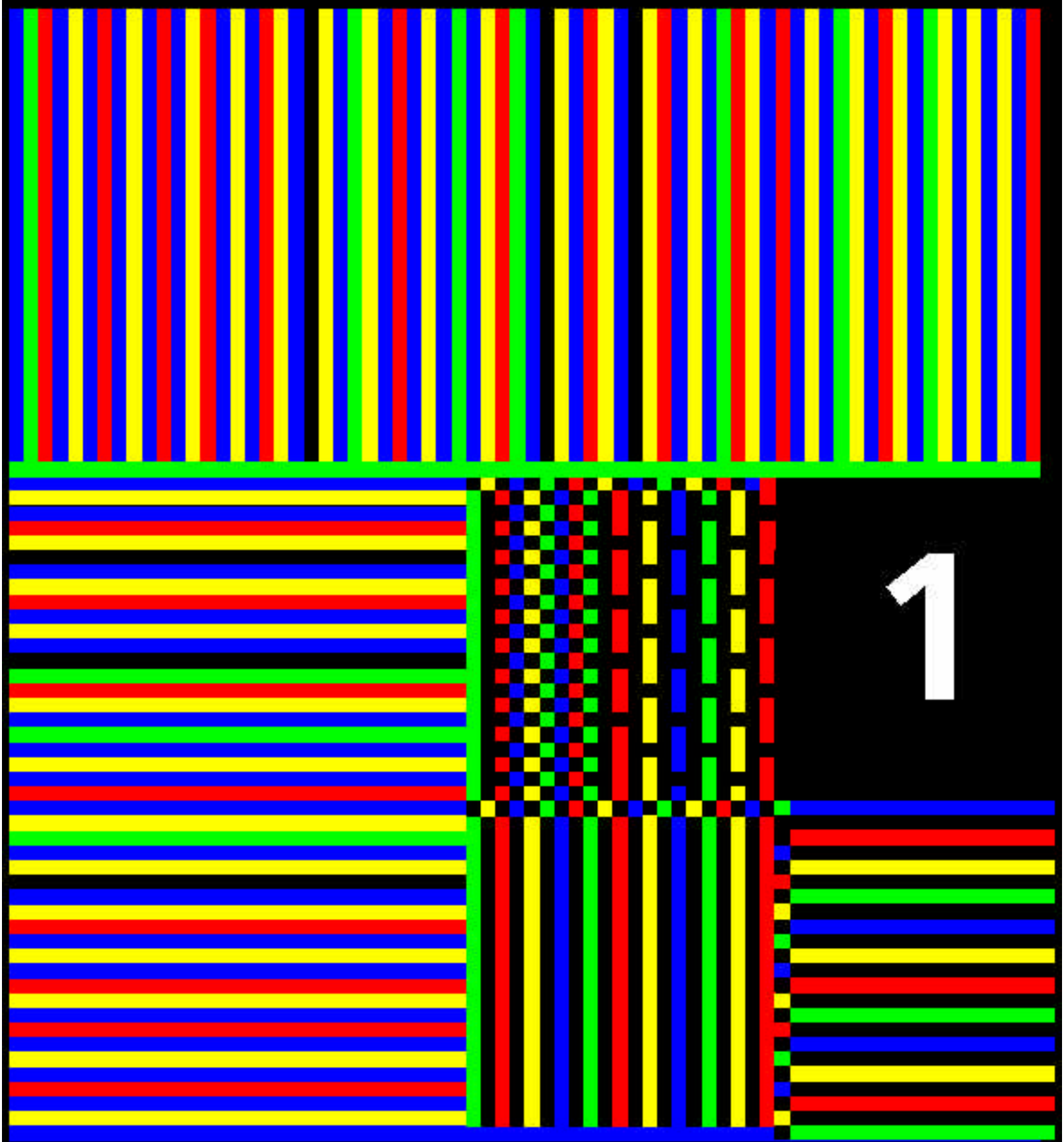
ISSN 1850-2954

IUNA

Instituto
Universitario
Nacional del
Arte

RIM

revista de investigación multimedia
año 1, número 1, invierno de 2006



editorial

Narciso y biombo:
Uno al otro ilumina
Blanco en lo blanco.

(Matsuo Basho, ca. 1650, Trad. Octavio Paz)

La multiplicidad, la aplicación de recursos tecnológicos y el entrelazamiento de géneros artísticos y comunicacionales son las marcas indudables de la Producción Multimedia. Al constituir una estrategia de comunicación e interacción apoyada en las tecnologías, su escenario inicial fue la pantalla de las computadoras. Actualmente, su ámbito no se limita a los modos de entrada-salida provistos por las tecnologías y puede extenderse a los que proveen muchos géneros artísticos, como las Artes de la Escena, Artes Sonoras, Artes del Movimiento y Audiovisuales. Es esta extensión la que deseamos explorar en esta revista que presentará principalmente las actividades de Investigación-Producción-Docencia de nuestra Area de Artes Multimediales.

Resulta significativo que "lo investigable" sea, en su acepción arcaica, aquello que no se puede hallar o descubrir y que, sin embargo, designe a lo que es, o a lo que puede ser objeto de investigación. El objetivo de la Investigación es hallar lo que no pudo ser hallado, encontrar lo que nadie antes encontró. La marca principal de la investigación es, entonces, la innovación en el sentido antes aludido. El polimorfismo de las Artes Multimediales estimula a que se pongan en juego variadas estrategias en la búsqueda de lo inhallable; ya sea al profundizar las particularidades de cada lenguaje, género, o medio aislado, o al estudiar sus interacciones.

Por un lado, dos trabajos como los de Martín Groisman ("Qué hay de nuevo en los Medios") y Matías Romero Costas ("Imprevisibilidad: conflicto y oportunidad. Nuevas interfaces para la creación de un nuevo tipo de relato") incitan a la reflexión sobre las particularidades del discurso multimedial. Otros se concentran en los recursos tecnológicos comprometidos en la producción, como el de Pablo Cetta ("Procesamiento en tiempo real de sonido e imagen con pd-gem") que presenta uno de los entornos más difundidos, versátiles y poderosos para el desarrollo de instalaciones de audio-video. El de Raúl Lacabanne ("Diseño de presentaciones multimedia dinámicas para el análisis de la música electroacústica") combina diversas alternativas de aplicación de software con estrategias de representación analítica de sonido y música electroacústica, o el de Mariano Cura ("Espacialización y refuerzo de sonido en vivo con sistemas multicanal") describe los recursos de espacialización de sonido usados en la presentación de diversas obras de Teatro Acústico. Otros trabajos, como el de Carmelo Saitta se vinculan con Multimedia en el sentido en que exploran la interacción de varios medios (banda sonora con imagen animada) desde una perspectiva analítica.

Este número se completa con las reseñas de Actividades Académicas de las Areas de Artes Audiovisuales y de Crítica de Arte, además, por supuesto, de nuestra Area.

Las Artes Multimediales constituyen un género inasible que se recrea constantemente a partir del diálogo entre sus actores y sus medios, y cuya presencia hemos querido plasmar significativamente en este primer número de RIM. Podríamos decir que, de forma análoga, cada artículo se abre a los otros como lo hace hacia cada medio desde su concepción particular.

Comité Editorial de RIM

staff

RIM es una publicación del área de Artes Multimediales del IUNA
Yatay 843, Ciudad Autónoma de Buenos Aires, República Argentina

ISSN 1850-2954
Impreso en Garbarino
Uspallata 833, Buenos Aires
agosto de 2006

índice

procesamiento en tiempo real de sonido e imagen con pd-gem

pablo cetta

Doctor en Música, en la especialidad Composición. Ha desarrollado una extensa labor como compositor, docente e investigador.. Ha recibido diversas becas y distinciones, entre ellas la Beca Antorchas, del LIPM y Fundación Rockefeller para realizar trabajos de composición en el CRCA (Universidad de California, San Diego), el Primer Premio en el Concurso Internacional de Bourges (Francia), el Premio Euphonies d´Or, el Premio Municipal de Música y el Segundo Premio Nacional.

Durante los últimos años han sido creadas diversas herramientas informáticas para el procesamiento en tiempo real, tanto de sonido como de imagen. Una de ellas es PD *(Pure Data)*, un entorno gráfico de programación desarrollado por Miller Puckette, para la generación y el tratamiento de audio digital.

El programa es de libre distribución y de código abierto, de modo que es posible examinar y ampliar su contenido, escrito en lenguaje C. Accedemos al programa a través de Internet, en la dirección: http://www-crca.ucsd.edu/~msp/software.html.

GEM *(Graphic Environment for Multimedia)*, escrito originalmente por Mark Danks, es una librería de objetos PD para la generación y procesamiento de imágenes en tiempo real, disponible en: http://www.danks.org/mark/. Tanto PD como GEM pueden ser utilizados en diversas plataformas, como Win 32, IRIX, Linux y OSX, con el propósito de crear aplicaciones multimediáticas. En este artículo vamos a analizar algunas características del sistema PD-GEM a través de ejemplos de aplicación.

La programación se realiza a partir de objetos y elementos de control, interconectables por medio de cables virtuales. En nuestro primer programa (figura 1a) observamos un objeto suma (+) al cual pasamos el primer término a través de un *number box*, y el segundo término como argumento, dentro del mismo objeto. El segundo ejemplo recibe, en cambio, el segundo término como parámetro, y la modificación del contenido del primer *number box* dispara la operación.

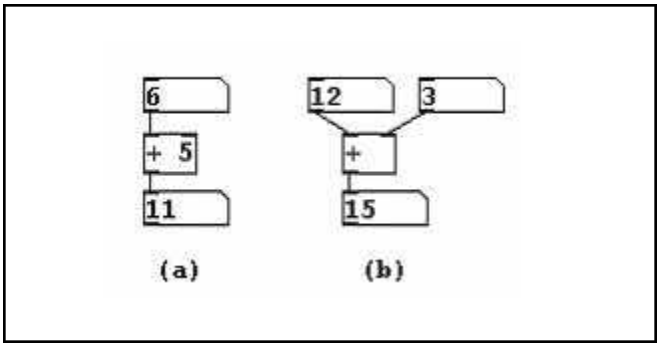


Figura 1

Otro modo de pasar información a un objeto es utilizando mensajes o listas de mensajes, y *sliders*, como se aprecia en la figura siguiente:

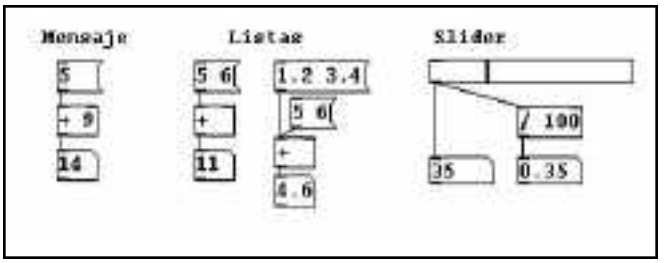


Figura 2

En estos ejemplos, las operaciones se disparan al hacer click con el mouse sobre el mensaje o la lista, pero también es posible "gatillar" un evento enviando un tipo de mensaje sin contenido alfanumérico, denominado *bang*. En el ejemplo que sigue, enviamos un *bang* –como mensaje o a través de un botón- a un objeto *random*, para que genere un número al azar comprendido entre o y 9. Si N es el argumento, los números generados pseudoaleatoriamente varían entre o y N-1. Es posible, además, retardar este tipo de mensaje utilizando el objeto *delay*, cuyo argumento es el tiempo de retardo en milisegundos. Los mensajes alfanuméricos, por otra parte, se retardan con *pipe*.

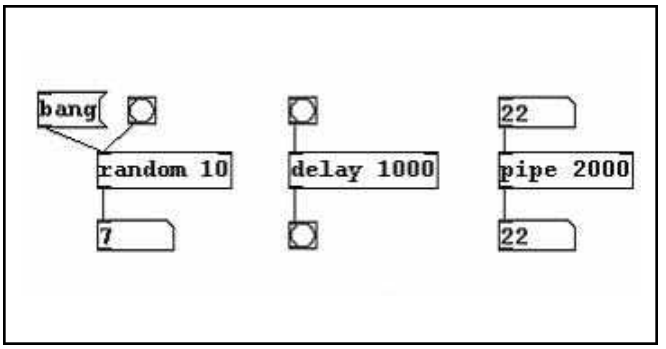


Figura 3

PD cuenta con una variada cantidad de objetos de síntesis y procesamiento, y con la posibilidad de incorporar objetos creados por terceros, denominados externals. Al igual que en Max-MSP, los nombres de objetos relacionados con señales de audio terminan con el símbolo "~". El siguiente gráfico muestra un ejemplo de síntesis por frecuencia modulada simple –se utilizan dos osciladores, conectados de tal modo que la salida del primero modifica la frecuencia del segundo– y el uso de filtros pasa altos, pasa bajos y pasa banda respectivamente, sobre un ruido blanco generado con *noise~*. El resultado de la síntesis es enviado al conversor digital-analógico, utilizando *dac~*.

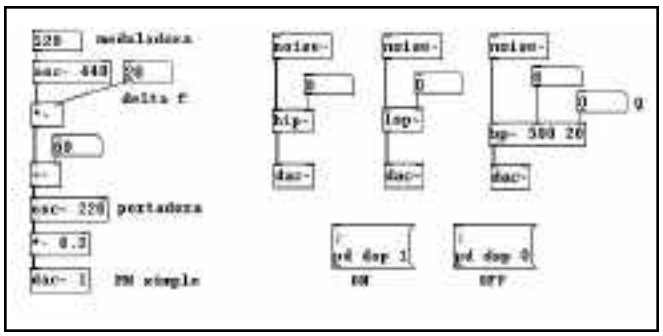


Figura 4

Según se muestra, tanto la frecuencia de los osciladores como las frecuencias de corte de los filtros, se especifican en Hertz como argumento, o como parámetro en las entradas a los objetos. Presionando el botón derecho del mouse sobre una unidad podemos visualizar su archivo de ayuda, donde se detallan las funciones de sus entradas y salidas, así como sus posibles argumentos. Vemos en la figura, además, dos mensajes destinados a iniciar y detener el procesamiento.

PD admite enviar datos usando nombres de variables y realizar conexiones de forma remota –sin la necesidad de cables. El ejemplo siguiente hace uso de estas posibilidades en mensajes de control y señales de audio.

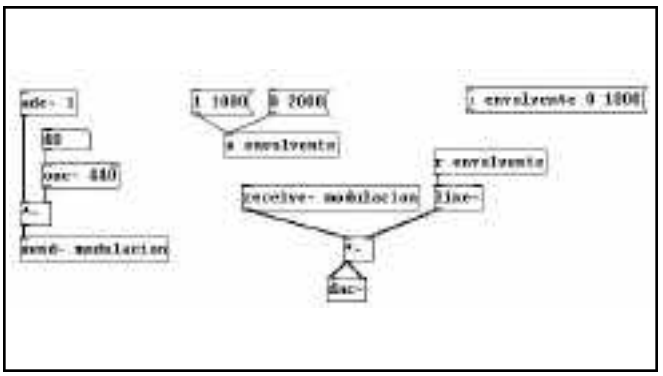


Figura 5

El objeto *adc~* de la figura 5 representa la salida del conversor analógico digital, al cual se encuentra conectado un micrófono. La señal que toma el micrófono es multiplicada luego por la función sinusoidal proveniente del oscilador, y se produce una "modulación en anillo". El resultado de la modulación es enviado de forma remota al objeto *dac~*. Por otra parte, la salida es multiplicada por una rampa (generada por *line~*) cuyos valores son recibidos a través de la variable envolvente. Cuando hacemos click sobre la primera lista, la señal cobra su máxima amplitud en 1000 milisegundos; la segunda lista, en cambio, produce una disminución gradual de 2 segundos. Por último, si hacemos click sobre el mensaje ubicado en el extremo superior derecho, enviamos los valores 0 y 1000 haciendo referencia al nombre de la variable. El punto y coma ubicado delante del nombre forma parte de la sintaxis.

Otra característica importante a considerar es la posibilidad de encapsular parte de un programa en un objeto del usuario. Denominamos *subpatch* al objeto del usuario que está contenido en el programa principal, y abstracción al subprograma que se guarda de forma independiente en el disco. Las abstracciones se invocan escribiendo el nombre del archivo en la caja de un objeto; los *subpatches* se crean con un nombre al que se antepone pd seguido por un espacio. Para determinar las entradas y salidas de un *subpatch* o abstracción utilizamos los objetos *inlet* y *outlet*. Veamos un ejemplo.

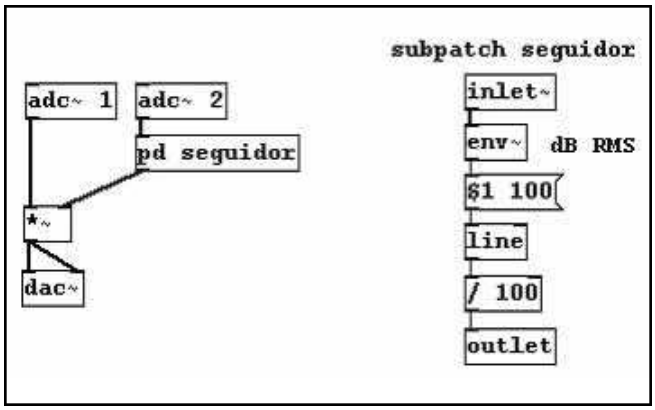


Figura 6

La figura 6, a la izquierda, representa un seguidor de envolvente. El programa extrae la evolución dinámica de la señal que ingresa por el micrófono conectado a la entrada 2 y la aplica a la señal que ingresa por la entrada 1. De este modo, controlamos la amplitud de una señal con otra. A la derecha se observa el contenido del *subpatch* denominado "seguidor", que se vale del objeto *env~* para calcular periódicamente la amplitud RMS en decibeles (o a 100). Con el propósito de crear una transición suave entre los valores, aplicamos una rampa de 100 milisegundos. Los valores devueltos por *env~* se cargan en la variable 1, que junto al número 100, conforman el mensaje destinado al objeto line.

En el próximo ejemplo trataremos de extraer la frecuencia fundamental de un sonido tónico, con el propósito de determinar su altura. Para esto, recurrimos a un external denominado *fiddle~*. En su aplicación más simple, obtenemos la frecuencia desde el *outlet* de la izquierda, en forma de nota MIDI (o a 127), la cual convertimos luego a un valor de frecuencia en Hertz con el objeto *mtof*.

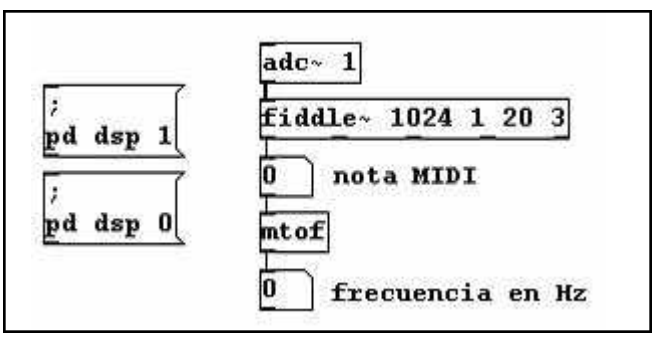


Figura 7

De este modo, pudimos extraer la intensidad y la altura de una señal entrante, valores que convenientemente utilizados pueden variar ciertos parámetros de una imagen en tiempo real, según veremos luego.

PD cuenta con poderosas herramientas que permiten graficar funciones. En el ejemplo que sigue, partimos de un banco de osciladores que genera los cinco primeros armónicos de un sonido complejo, sobre una fundamental de 172.2 Hz. Empleamos el objeto *table* para representar la forma de onda y el espectro –a través de un *subpatch* que aplica la transformada rápida de Fourier.

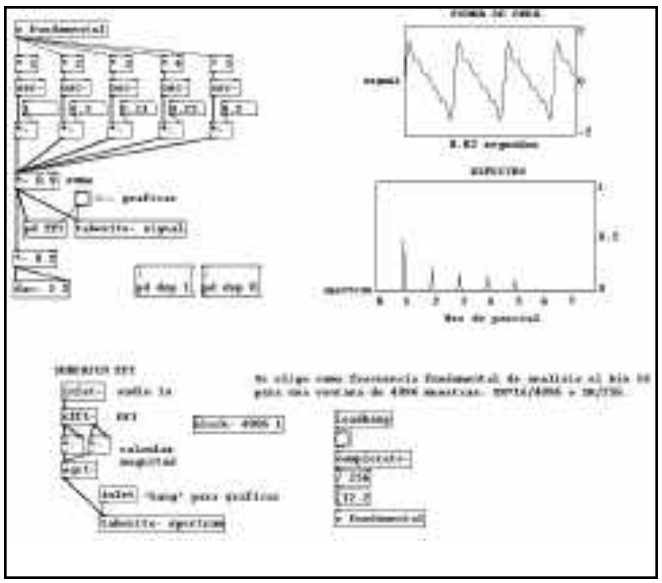


Figura 8

Graficamos la forma de onda enviando directamente la señal a través de un objeto *tabwrite~*. La unidad *rfft~* calcula la transformada de Fourier, donde cada muestra del espectro es representada por un número real (a) y otro imaginario (b). La magnitud de cada par de valores se obtiene a través de la raíz cuadrada *(sqrt~)* de "a" al cuadrado más "b" al cuadrado.

La transformada de Fourier en audio digital se utiliza en distintos tipos de procesamiento. Encontramos un ejemplo de aplicación en la convolución simple, proceso en el cual extraemos la envolvente espectral de un sonido y la aplicamos a otro sonido. A la izquierda de la figura siguiente observamos el cuerpo principal del programa. Con objetos *tabplay~* ejecutamos dos archivos de audio almacenados en el disco rígido, cuyos contenidos ingresan luego al *subpatch* de convolución. El objeto *soundfiler* recibe un mensaje sobre el nombre y ubicación del archivo, y sobre la denominación de una tabla que sirve de soporte a sus muestras. El objeto *tabplay~* lee esas muestras de acuerdo a la frecuencia de muestreo. Con el propósito de mejorar los resultados vamos a utilizar una ventana *Hanning* en el cálculo de la *FFT*, en lugar de una ventana rectangular, la generación y almacenamiento de la función se aprecia en el sector derecho del gráfico.

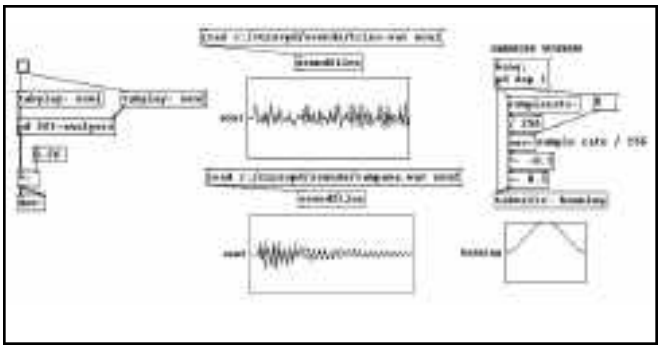


Figura 9

La convolución ocurre dentro del *subpatch* denominado *fft-analysis* (figura 10). Cada bloque de datos –en este caso fijado en 256 muestras– es multiplicado por la función *Hanning* y luego transformado al dominio de la frecuencia con *rfft~*. De la señal de la derecha se extraen los valores de amplitud para cada muestra del espectro y luego se multiplican por los de la señal de la izquierda. De este modo, la resultante contiene las componentes de frecuencia del primer sonido, pero con las variaciones de amplitud del segundo. Finalmente, se normalizan los valores a 1/256 y se envían al objeto que calcula la *FFT* inversa, para regresar al dominio del tiempo.

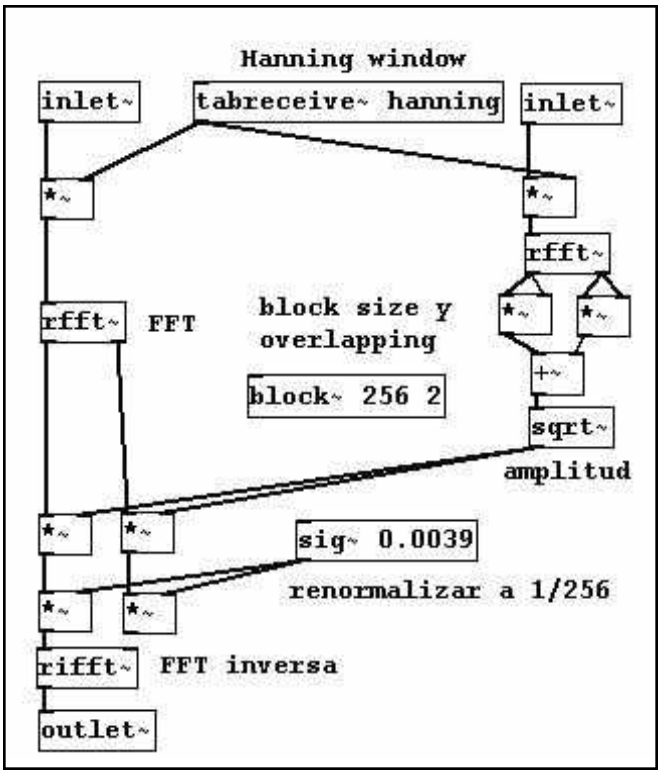


Figura 10

Para el tratamiento de la imagen contamos con una gran variedad de objetos pertenecientes a la librería GEM. Para comenzar, vamos a crear un ejemplo muy simple en el cual dibujamos un cuadrado, le aplicamos color y lo rotamos en el espacio.

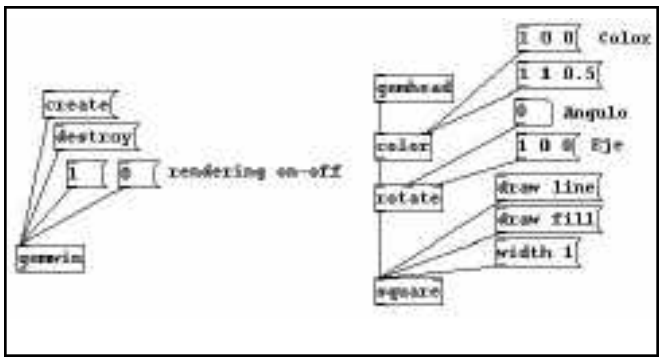


Figura 11

El objeto *gemwin* crea, al recibir el mensaje create, la ventana donde dibujamos; y al recibir un 1 o un 0 inicia o detiene el procesamiento gráfico. El programa comienza siempre con el objeto *gemhead*, al que siguen unidades modificadoras de generación de formas geométricas, de representación de imágenes, etc. En nuestro ejemplo utilizamos un objeto *color* y otro *rotate* para cambiar la apariencia del cuadrado *(square)*. Vemos que *color* acepta una lista con valores RGB, *rotate* un ángulo y uno o más ejes sobre los cuales rotar, y *square* mensajes que definen la figura llena o simplemente el contorno y su espesor.

Existen objetos para la representación de diversas formas geométricas. En el ejemplo siguiente dibujamos una esfera, a la que aplicamos iluminación local.

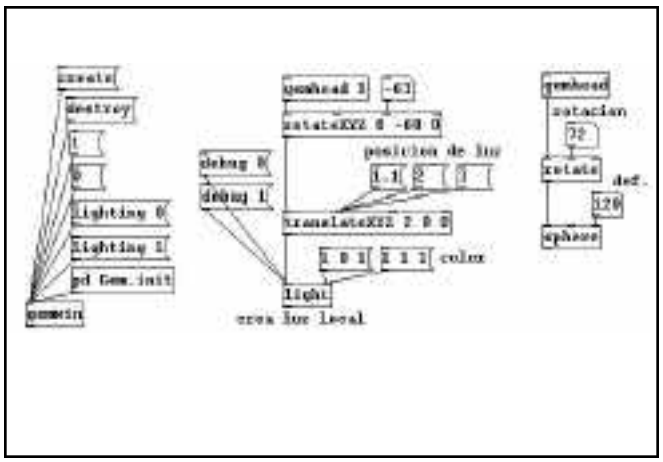


Figura 12

La iluminación se activa o desactiva a través del mensaje *lighting*, enviado a *gemwin*. La unidad *light* crea iluminación local –que es diferente a la iluminación global, producida con *world_light*– y admite cambios de color y posicionamiento en el espacio. El mensaje *debug* permite visualizar la ubicación de la fuente. El tercer parámetro del objeto *sphere* indica el nivel de definición del gráfico.

Veamos ahora la utilización de un archivo gráfico como textura de las caras de un cubo. Empleamos para ello los objetos *pix_image y pix_texture*.

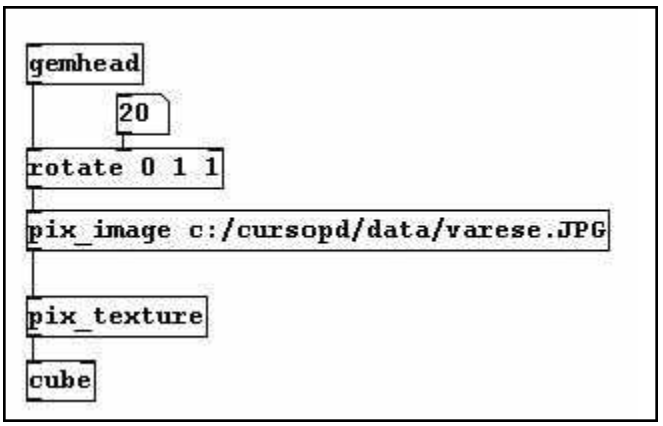


Figura 13

En el ejemplo siguiente vamos a combinar sonido e imagen en sincronismo. Partimos de una fuente sonora virtual que rota alrededor del oyente, simulada mediante la reproducción a través de cuatro parlantes. Para ello, utilizamos una técnica de localización espacial denominada *Ambisonics*. De forma simultánea, visualizamos una esfera inmóvil que representa al oyente y otra que gira en círculos alrededor de la primera, y representa a la fuente.

Ambisonics consta de un proceso de codificación de la señal a espacializar, y de otro proceso de decodificación, de acuerdo a la cantidad de parlantes a utilizar y su disposición en el espacio de audición. En nuestro caso –cuatro parlantes dispuestos en un cuadrado– las ecuaciones de codificación son tres: W, X e Y.

$$w(n) = 0.707\ s(n) / d \qquad (W)$$
$$x(n) = \sin q\ \ s(n) / d2 \qquad (X)$$
$$y(n) = \cos q\ \ s(n) / d2 \qquad (Y)$$

donde s(n) es la señal monofónica a espacializar, q es el ángulo de posicionamiento de la fuente medido sobre el plano horizontal, y d es la distancia. La decodificación de las señales destinadas a cada parlante se logra por medio de:

$$s1(n) = W + X + Y$$
$$s2(n) = W + X - Y$$
$$s3(n) = W - X - Y$$
$$s4(n) = W - X + Y$$

Lo visto –codificación y decodificación– se resume en el *subpatch* siguiente. Por el *inlet~* ingresa la señal de audio a espacializar, la variable *ang* recibe el ángulo de rotación de la fuente en grados, y *a_dist* la distancia. Es importante mantener esta última distinta de cero para evitar la división ilegal por ese valor. Como los objetos *sin* y *cos* sólo admiten ángulos en radianes, se realiza la conversión multiplicando por 2p/360 (0.017453).

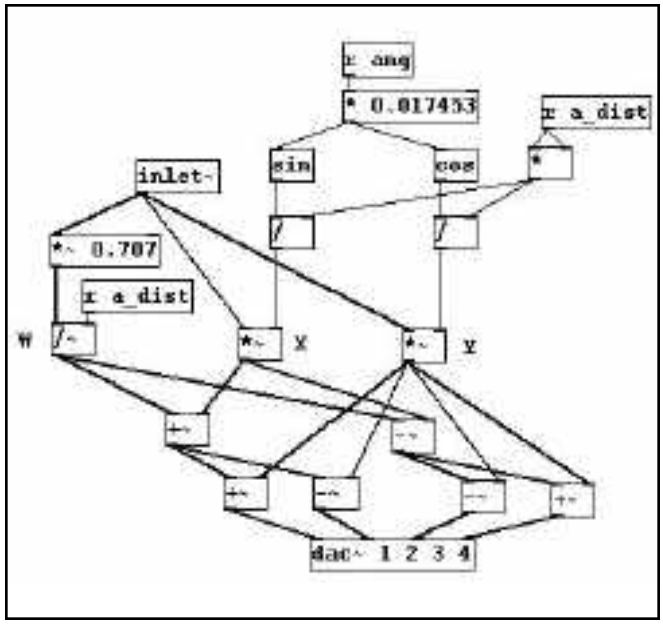


Figura 14

En la figura 15 vemos el *patch* principal. El objeto *counter* genera cíclicamente números entre 0 y 359, incrementando una unidad cada vez que recibe un bang desde metro, y los valores se envían a través de la variable *ang*. El *subpatch* denominado generador produce repetidamente una banda de ruido fácilmente localizable, mientras que la espacialización se resuelve en el *subpatch* ambisonics, explicado anteriormente. Finalmente, dibujamos dos esferas, una fija que representa al oyente, y otra que rota, desplazada según la distancia fuente-oyente.

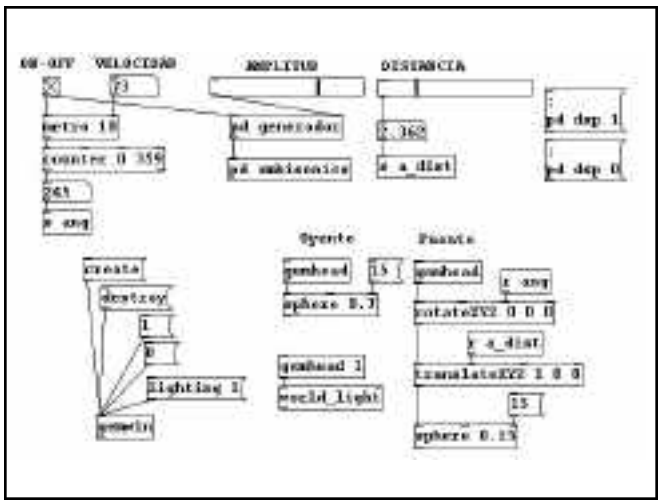


Figura 15

Para nuestro último ejemplo vamos a utilizar la altura y la intensidad de los sonidos de un instrumento musical para aplicarlos en el procesamiento de una imagen. La altura va a controlar el ángulo de posicionamiento de una fuente de iluminación local y la dinámica va a regular el nivel de transparencia de una esfera. Dentro de ésta ubicaremos otra de menor tamaño, en rotación sobre sí misma, sobre la cual proyectaremos un video.

Suponiendo que sólo ejecutamos notas de la octava central (notas MIDI 60 a 72) realizamos primero un escalamiento para adaptar los valores a un rango de 0 a 360. Para ello, restamos 60 al número de nota detectado y luego multiplicamos por 30. El *subpatch* para la detección de la amplitud, visto anteriormente, nos devuelve valores comprendidos entre 0 y 1, lo cual no requiere escalamiento alguno. En la parte inferior de la figura 16 vemos el *subpatch* seguidor y el external *fiddle~*, que envían los valores detectados a través de las variables dinámica y altura.

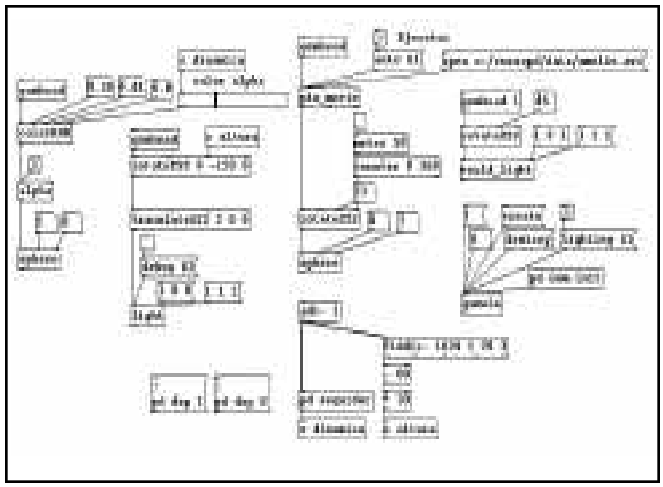


Figura 16

La variable dinámica ingresa de forma remota al objeto *colorRGB*. El cuarto *inlet* es el que corresponde al nivel de transparencia, pero siempre en combinación con la unidad *alpha* conectada más abajo. En la segunda columna de la figura controlamos la iluminación local, que rota de acuerdo a la altura. A la derecha, encontramos el objeto *pix_movie*, que proyecta el video sobre la esfera interior, que rota un grado cada vez que recibe un *bang* desde metro. Por último, más a la derecha, se aprecia un objeto de iluminación global, cuya fuente puede rotar manualmente de acuerdo al número ingresado.

Hemos visto, a través de estos ejemplos, algunas aplicaciones del sistema PD-GEM. La lista de objetos es extensa y continúa en expansión, sin embargo, utilizando unos pocos de forma apropiada, fue posible lograr resultados interesantes. El empleo de controladores MIDI, cámaras de video u otros dispositivos de ingreso de datos aportan múltiples posibilidades de interacción; y la conexión de dos o más computadoras en red aumenta considerablemente la capacidad de procesamiento para la realización de programas complejos.

Bibliografía

Puckette, M. PD documentation. http://www.crca.ucsd.edu/~msp/software.html.

Danks, M. GEM Manual. http://gem.iem.at/

Malham, D.G. Spatial Hearing Mechanisms And Sound Reproduction. University of York, England, 1998. http://www.york.ac.uk/inst/mustech/mtg.htm